Conventions de représentation d'un DTD dans le modèle GER

Delcroix, Christine

Publication date:
2001

Facultés Universitaires Notre-Dame de la Paix
Namur - Belgium

Conventions de représentation d'un DTD dans le modèle GER

Le modèle **E**ntité **A**ssociation **E**tendu[1] permet la représentation de structure de données issues de multiples paradigmes (relationnel, système de fichiers, orienté objet, hiérarchique, ...) à différents niveaux d'abstraction (physique, logique ou conceptuel). On peut donc envisager la représentation de la structure de données XML dans ce même modèle. Cette structure peut être décrite selon plusieurs mécanismes établis par le consortium W3C, responsable du développement de XML. C'est au premier né, le fichier de **D**éfinition de **T**ype de **D**ocument[2] que nous nous sommes intéressés.

Ce document présente les conventions de représentation d'un DTD dans le modèle GER.

Côté outil, le modèle GER trouve une implémentation complète dans l'atelier DB-Main. De plus, l'atelier est maintenant enrichi d'un extracteur XML permettant d'établir automatiquement la représentation dans le modèle GER du DTD en entrée.

1. En anglais, il est connu sous le nom **G**eneric **E**ntity **R**elationship Model (G.E.R.).
2.En anglais, il est désigné par le terme **D**ocument **T**ype **D**efinition et est abrégé de la même façon qu'en français par les initiales D.T.D.

1. Un rapide coup d'oeil

Cette première section donne une vue d'ensemble de la représentation d'un DTD sous la forme d'un schéma du modèle GER. La figure 1.1 illustre un DTD décrivant la structure de rapports formatés en XML. La figure 1.2 illustre la représentation graphique de ce même DTD dans le modèle GER.

```
<!ELEMENT Employment (Job | Employer | JobSeeker | Resume)*>

<!ELEMENT Job (Offer, Requirement*, PublicationDate?)>

<!ATTLIST Job JobID ID #REQUIRED
              Employer IDREF #REQUIRED
              OriginalDocument ENTITY #IMPLIED>

<!ELEMENT Employer (Name, (Mail|Phone|Fax)+ , Activity*, ApplyProc)>

<!ATTLIST Employer EmployerId ID #REQUIRED
                   Jobs IDREFS #IMPLIED
                   WebSite ENTITY #IMPLIED>

<!ELEMENT Activity (#PCDATA)>
<!ELEMENT ApplyProc (#PCDATA)>
<!ELEMENT Fax (#PCDATA)>
<!ELEMENT JobSeeker (#PCDATA)>
<!ELEMENT Mail (#PCDATA)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Offer (#PCDATA)>
```

Fig. 1.1- DTD lié au monde de l'emploi.

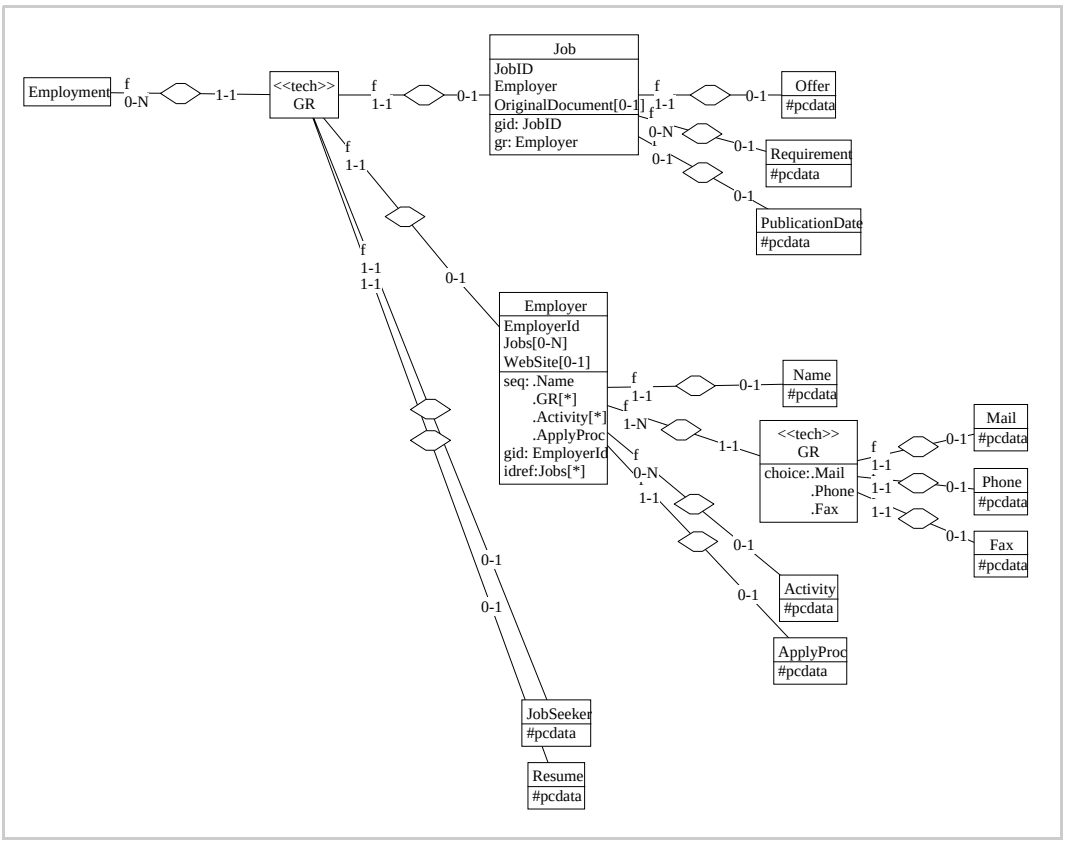

Fig. 1.2- Représentation graphique de la DTD (cf. fig. 1.1)

Les règles de représentation seront détaillées dans les sections suivantes. Nous avons isolé dans chaque section, un ou plusieurs concepts du DTD :

- représentation d'un type d'éléments et de son contenu : section 2;
- représentation d'une déclaration de liste d'attributs : section 3;
- représentation d'entité et de notation : section 4.

La section 5 décrit, quant à elle, les conditions que doit vérifier une DTD pour être représentable sous la forme d'un schéma du modèle GER selon les conventions décrites dans ce document.

2. Représentation d'un type d'éléments

Chaque type d'éléments est représenté par un type d'entités de même nom. Il existe quatre types de contenu différents, chacun d'eux ayant sa propre représentation graphique.

2.1. Type de contenu ANY

Le type de contenu ANY est représenté par un attribut simple nommé #any, attaché au type d'entités représentant ce type d'éléments. La figure 2.1 illustre cette règle pour la déclaration suivante :

```
<!ELEMENT JobSeeker ANY>
```

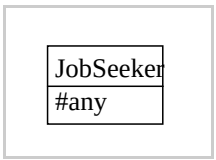

Figure 2.1- Représentation graphique du contenu ANY.

2.2. Type de contenu EMPTY

Un type d'entités correspondant à un type d'éléments de contenu EMPTY ne possède aucune construction représentant son contenu. C'est précisément l'absence de construction qui caractérise la représentation d'un tel type de contenu. La figure 2.2 illustre cette règle pour la représentation suivante :

```
<!ELEMENT Employer EMPTY>
```

Figure 2.2- Représentation graphique du contenu EMPTY.

2.3. Type de contenu ELEMENT

Considérons, par exemple, la déclaration suivante :

```
<!ELEMENT Employer (Name, (Mail|Phone|Fax)+ , Activity*, ApplyProc)*>
```

Terminologie. Si on analyse le contenu d'un type d'éléments de contenu ELEMENT, on constate qu'il est formé d'un certain nombre de constituants. Chaque constituant est soit

- un type d'éléments, éventuellement soumis à un opérateur de cardinalité (?, *, +). Dans ce cas, on parle de **constituant élémentaire**.
- lui-même constitué de plusieurs constituants regroupés au sein de parenthèses. Dans ce cas, on parle de constituant composite ou **groupe**. Un groupe peut également être soumis à un opérateur de cardinalité.

Dans la suite de ce document, nous désignerons par groupe, tous les regroupements délimités par des parenthèses, **y compris les parenthèses les plus extérieures quand elles sont soumises à l'action d'un opérateur de cardinalité**.

Ainsi, dans l'exemple ci-dessus, le contenu du type d'éléments Employer est formé d'un groupe (correspondant aux parenthèses les plus extérieures car elles sont soumises à l'action de l'opérateur de cardinalité *). Ce groupe est formé successivement d'un constituant élémentaire Name,

d'un groupe `Mail-Phone-Fax`, suivi de deux constituants élémentaires `Activity` et `ApplyProc`

Par contre, dans la déclaration suivante :

```
<!ELEMENT Employer2 (Name, (Mail|Phone|Fax)+ , Activity*, ApplyProc)>
```

Le contenu du type d'éléments `Employer2` est formé de quatre constituants : un constituant élémentaire `Name`, un groupe `Mail-Phone-Fax` et deux constituants élémentaires `Activity` et `Apply-Proc`. Il n'y a donc pas de groupe associé aux parenthèses extérieures car elles ne sont pas soumises à l'action d'un opérateur de cardinalité.

Bien que cette terminologie peut paraître étrange, elle trouve sa justification dans le fait que les parenthèses utilisées dans le contenu d'un type d'éléments ont deux rôles :

- délimiter la portée de l'opérateur de cardinalité. Les constituants concernés forment alors un *groupe*;
- séparer le nom du type d'éléments de la définition de son contenu. C'est le cas des parenthèses les plus extérieures.

Chaque constituant (élémentaire ou groupe) peut être soumis à un **opérateur de cardinalité** (`? *`, `+`).

A l'intérieur de parenthèses, les constituants peuvent être ordonné et former une **séquence**. Dans ce cas, ils sont séparés par une virgule. Ils peuvent également constituer les diverses alternatives d'un **choix**. Dans ce cas, ils sont séparés par une barre verticale. Par exemple, les constituants du contenu du type d'éléments `Employer` constituent une séquence; alors que ceux du groupe `Mail-Phone-Fax` constituent un choix.

Enfin, nous désignons par **lien père-fils**, le lien qui existe entre un type d'éléments (appelé père) et son contenu (appelé fils) ou entre un groupe (appelé père) et ses différents constituants (appelés fils).

La suite de cette section donne la succession des règles de représentation d'un type d'éléments de contenu `ELEMENT`. Ces règles seront illustrées sur l'exemple du type d'éléments `Employer`.

Règle 1. Le type d'éléments déclaré est représenté par un type d'entités de même nom.

Règle 2. Chaque constituant (élémentaire ou groupe) du contenu du type d'éléments est représenté par un type d'entités. Si le constituant est

- élémentaire, le type d'entités qui le représente porte le même nom.
- composite, le type d'entités qui le représente porte un nom quelconque et est soumis au stéréotype <<tech>>.[1]

Ainsi, le type d'éléments `Employer` est représenté par un type d'entités de même nom. Son contenu est un groupe représenté par un type d'entités technique (appelé `GR`). Ce groupe a quatre

1. Pour associer un sttype d'associationsréotype à un type d'entités dans l'outil DB-Main, il suffit d'accéder à la boîte de dialogue *Prop* de ce type d'entités, de sélectionner la méta-propriété *Stereotype* et de lui ajouter la valeur prédéfinie *tech* apparaissant dans le cadre de droite. L'ajout de valeurs prédéfinies telles que *tech* se fait via le menu *Product/Meta/Properties...*, sélectionnez le méta-objet *Entity Type* et la propriété dynamique *Stereotype*, cliquez sur le bouton *Modify*, puis sur *Sem*. Le champs sémantique doit contenir les caractères suivants :
   #values=
   valeur_prédéfinie_1
   valeur_prédéfinie_2
   ...
   #

constituants :

- 3 constituants élémentaires (`Name`, `Activity` et `ApplyProc`), représentés, tous trois, par un type d'entités de même nom;
- un groupe de trois constituants élémentaires, représenté par un type d'entités technique dont le nom peut être librement choisi (ici, `GR1`). Les trois constituants de ce groupe (`Mail`, `Phone` et `Fax`) donnent lieu à trois types d'entité de même nom.

La figure 2.3 montre la représentation (en cours de construction) du type d'éléments `Employer` et de son contenu.

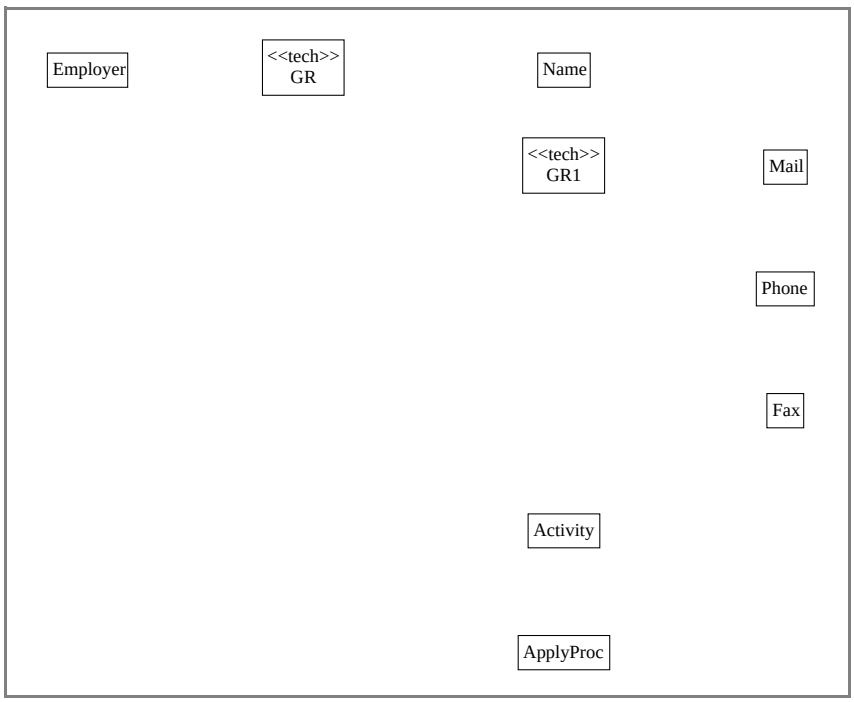

Figure 2.3- Chaque constituant est représenté par un type d'entités.

Règle 3. Chaque lien père-fils est représenté par un type d'associations. Le nom du type d'associations est laissé au libre choix. Cependant, nous recommandons de le préfixer par une barre verticale de façon à ce qu'il ne soit pas apparent.

Dans un type d'associations, la cardinalité du rôle joué par le fils vaut toujours `[1-1]` sauf si le fils a plusieurs pères, auquel cas la cardinalité vaut `[0-1]`. Nous renvoyons à la section suivante pour une justification de ce choix.

Quant à la cardinalité du rôle joué par le père, elle dépend de l'opérateur de cardinalité auquel est soumis le constituant correspondant au fils dans le DTD :

- si l'opérateur de cardinalité est `?`, la cardinalité du rôle est `[0-1]`;
- si l'opérateur de cardinalité est `*`, la cardinalité du rôle est `[0-N]`;
- si l'opérateur de cardinalité est `+`, la cardinalité du rôle est `[1-N]`;
- si il n'y a pas d'opérateur de cardinalité, la cardinalité du rôle est `[1-1]`;

Dans un type d'associations, le rôle joué par le père doit être nommé `f` (abrégé de `father`). Cette information permettra d'identifier le type d'entités père dans certains cas limites.

Enfin, il reste à ajouter deux types d'information afin que la représentation soit complète. En effet,

à ce stade-ci, ni l'ordre des constituants, ni le type de séparateur de ces constituants n'apparaît dans le schéma. Pour ce faire, pour chaque type d'entités **qui a plus d'un fils**, on définit un groupe constitué des rôles que ses fils jouent avec lui et soumis à unecontrainte spécifique : `seq` ou `choice` indiquant le type de séparateur. L'ordre des rôles dans le groupe représente l'ordre des constituatns dans le DTD.

La figure 2.4 illustre l'application de la règle 3 pour le type d'éléments `Employer`.

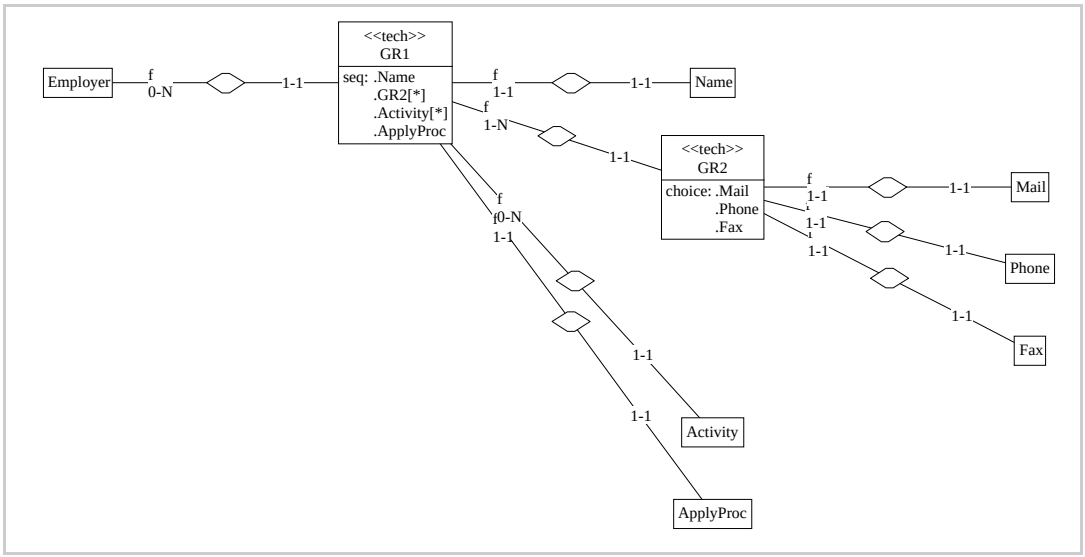

Figure 2.4- Les liens père-fils sont modélisés par des types d'association.

Le lien entre le type d'entités `Employer` et le groupe correspondant à son contenu est modélisé par un type d'associations joignant le type d'entités `Employer` au type d'entités technique `GR` représentant ce groupe.

Le rôle joué par `Employer` a la cardinalité `[0-N]` puisque le groupe correspondant aux parenthèses extérieures est soumis à l'opérateur de cardinalité `*`. Le rôle joué par `GR` vaut `[1-1]` car `GR` n'a qu'un seul père.

`GR` a quatre fils auxquels il est lié par le biais de types d'association et correspondant aux quatre constituants qui le forment. Seuls ses deuxième et troisième constituants sont soumis à un opérateur de cardinalité (resp. `+` et `*`). Les rôles joués par `GR` auront donc les cardinalités suivantes : `[1-1]`, `[1-N]`, `[0-N]` et `[1-1]`.

Ces quatre constituants sont séparés par des virgules. `GR` contient donc un groupe soumis à la contrainte `seq` et constitué des quatre rôles joués par ces fils. L'ordre de ces quatre rôles dans le groupe reflète l'ordre des quatre constituants correspondants dans le DTD.[1]

2.4. Type de contenu `MIXED`

Le type de contenu `MIXED` suit des règles similaires à celles décrites à la section précédente pour le cas du type de contenu `ELEMENT`. Cependant, ces dernières doivent être complétées afin de représenter le terme `#PCDATA` pouvant apparaître ici.

1. On voit mieux, maintenant, pourquoi il était nécessaire d'associer un groupe aux parenthèses extérieures du contenu d'un type d'éléments, uniquement lorsque celles-ci étaient soumises à l'action d'un opérateur de cardinalité. En effet, dans ce cas, il fallait un rôle permettant de représenter cet opérateur de cardinalité. L'ajout du type d'entités technique correspondant à ces parenthèses a pour conséquence l'ajout d'un type d'associations (et donc de deux rôles!).

```
<!ELEMENT Contact (#PCDATA|Mail|Phone|Fax)*>
```

Le terme `#PCDATA` est représenté par un attribut simple nommé `#pcdata`. Cet attribut est inclus dans le groupe `seq` ou `choice` au même titre que les rôles joués par les fils, puisqu'il est un constituant du groupe correspondant aux parenthèses extérieures au même titre que les types d'élément `Mail`, `Phone` et `Fax`.

La figure 2.5 présente le type d'éléments `Contact` et son contenu de type `MIXED`.

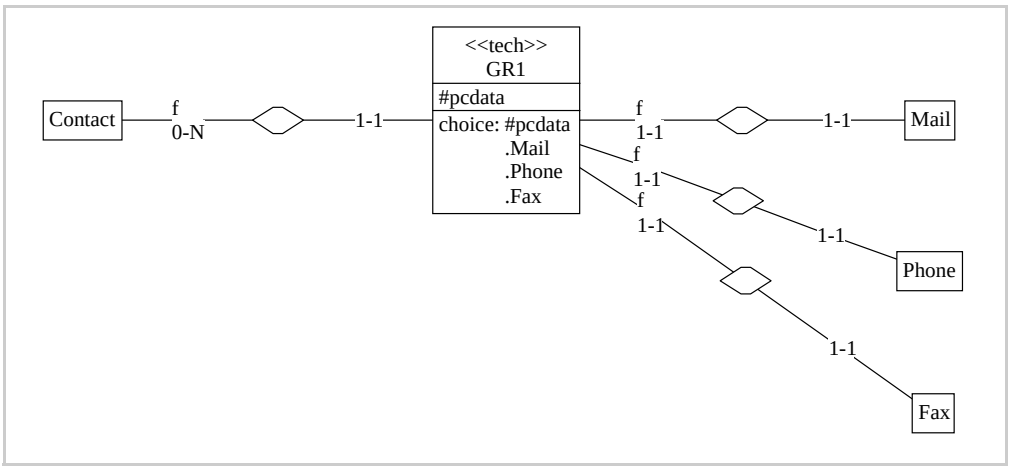

Figure 2.5- Le constituant `#PCDATA` est représenté par un attribut `#pcdata`.

Avec ces conventions, un type d'éléments qui a un contenu `#PCDATA` simple est représenté par un type d'entités contenant un seul attribut nommé `#pcdata`. Puisque ce type d'éléments n'a qu'un seul fils, il n'est pas nécessaire de créer de groupe `seq` ou `choice`. Par exemple, le type d'éléments `Name` dont la déclaration figure ci-dessous est représenté à la figure 2.6.

```
<!ELEMENT Name (#PCDATA)>
```

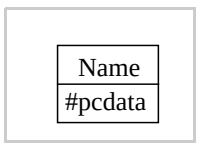

Figure 2.6- Représentation d'un type d'éléments de contenu `#PCDATA`.

3. Justification de certains choix de représentation peu intuitifs ...

Dans cette section, nous allons développer deux points qui nous semble importants. Le premier traite de la cardinalité des rôles joués par les fils qui tantôt vaut `[1-1]`, tantôt vaut `[0-1]`. Le second s'attache à expliciter la sémantique du type de contenu `"choice"`.

3.1. Cardinalité des rôles joués par les fils

Chaque type d'éléments du DTD est représenté par un type d'entités. Au niveau des données, chaque élément d'un document XML est représenté par une instance du type d'entités correspondant à son type. Ceci implique que l'on peut trouver, dans un type d'entités, deux instances identiques pour peu que le document XML contenaient deux éléments de même type et de même contenu. Ce choix de représentation s'apparente à une représentation par instances (à opposer à une représentation par valeurs).

La nature arborescente d'un document XML nous permet d'affirmer que chaque élément d'un document XML a un et un seul élément parent (sauf s'il était la racine). Du point de vue de la représentation adoptée, cela signifie que chaque entité (sauf du type de la racine) est liée à une et une seule entité d'un type parent. Nous pouvons alors distinguer deux situations différentes :

- le type d'entités a un seul type d'entités parent. Dans ce cas, la cardinalité du rôle qu'il joue dans le type d'associations le joignant à son père vaut `[1-1]`.

  Par exemple, soit le DTD complet suivant :

```
<!ELEMENT Employment (Job | Employer | JobSeeker | Resume)>
<!ELEMENT Job (#PCDATA)>
<!ELEMENT Employer (#PCDATA)>
<!ELEMENT JobSeeker (#PCDATA)>
<!ELEMENT Resume (#PCDATA)>
```

  Dans ce cas, les cardinalités des rôles des types d'entité `Job`, `Employer`, `JobSeeker` et `Resume` en tant que fils du type d'entités `Employment` valent toutes [1-1], signifiant que chaque élément du type `Job` est le fils d'un et un seul élément de type `Employement`. La figure 3.1 montre cette représentation.

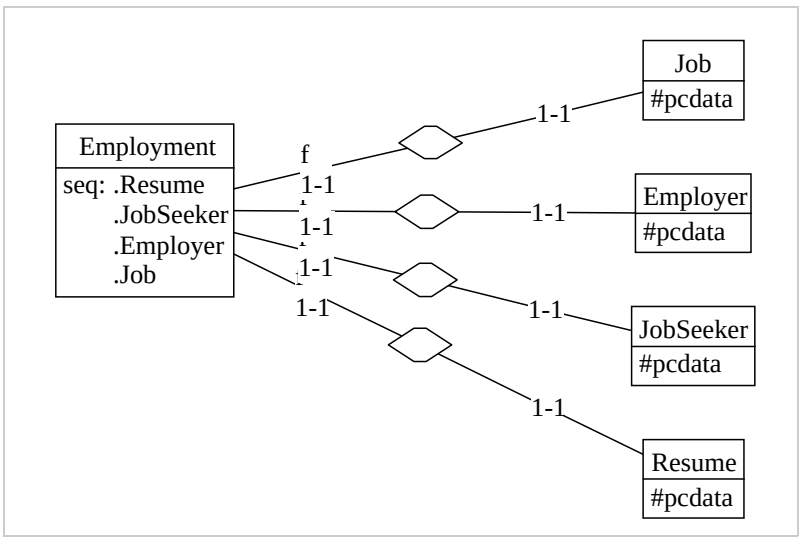

Figure 3.1- La cardinalité des rôles joués par les fils n'ayant qu'un seul père doit valoir [1-1].

- le type d'entités a plusieurs types d'entité parents. Dans ce cas, à chaque entité de ce type correspond une et une seule entité d'un des types parents. La cardinalité des rôles que le type d'entités joue dans les types d'association le joignant à ses pères vaut `[0-1]`. De plus, on peut exprimer le fait que chaque entité joue un rôle dans un seul des types d'association les joignant à son père par le recours à une contrainte `exactly-1` définie sur les rôles fils.

  Par exemple, soit le DTD suivant :

```
<!ELEMENT Employer (Name)>
<!ELEMENT JobSeeker (Name)>
<!ELEMENT Firm (Name)>
```

  Dans ce cas, les cardinalités des rôles joués par le type d'entités `Name` sont toutes égales à `[0-1]`. La contrainte `exactly-1` définies sur les rôles exprime que chaque entité `Name` a une seule entité parent de type `Employer`, `JobSeeker` ou `Firm`. La figure 3.2 montre la représentation graphique de ce DTD.

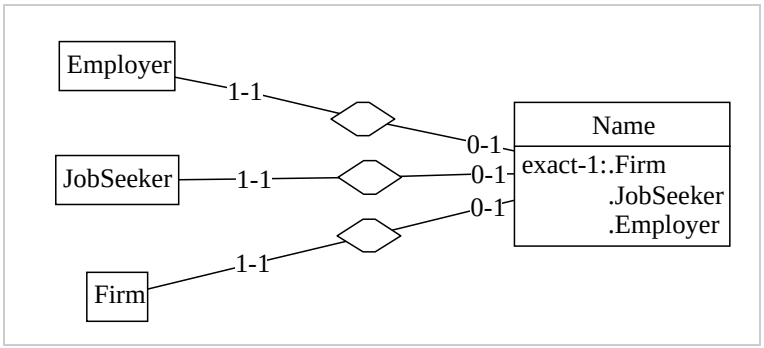

Figure 3.2- Chaque entité `Name` est associée a une et une seule entité "père" du type `Employer`, `JobSeeker` ou `Firm`.

3.2. Groupe `choice`

Soit la déclaration de type d'éléments suivante :

```
<!ELEMENT Contact (Mail|Phone|Fax)>
```

L'utilisation de la barre verticale en guise de séparateur signifie que chaque élément `Contact` sera constitué au choix d'un élément `Mail`, d'un élément `Phone` ou d'un élément `Fax`. Dans le modèle Entité-Association Etendu, nous disposons de la contrainte d'exclusion pour exprime un tel fait. Le recours à une telle contrainte nous mène au schéma suivant :

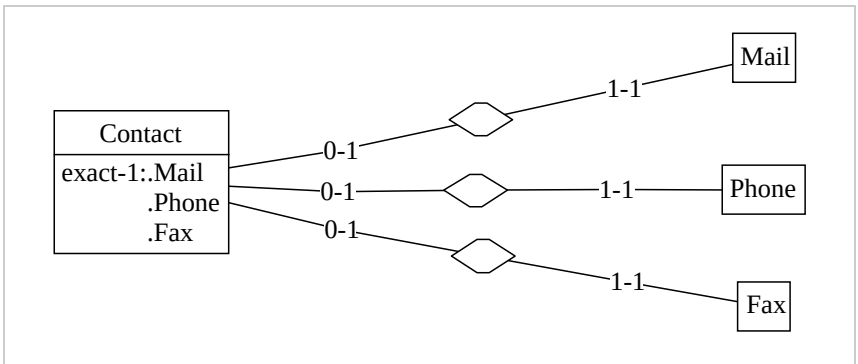

Figure 3.3- Les constituants forment un contenu de type `choice` et ne sont pas soumis à un opérateur de cardinalité.

Cette représentation semble bien adaptée à la déclaration énoncée ci-dessus. Mais si un des constituants est soumis à un opérateur de cardinalité, celui-ci ne peut être représenté dans le schéma. En effet, l'utilisation de la contrainte d'exclusion nous force à travailler avec des rôles facultatifs. La cardinalité des rôles père ne peut donc plus être utilisée pour représenter l'opérateur de cardinalité auquel est soumis le fils.

Considérons le DTD suivant :

```
<!ELEMENT Contact (Mail*|Phone|Fax+)>
```

Il peut être représenté en utilisant des types d'entité techniques, comme le montre la figure 3.4. Ces types d'entité techniques n'ont qu'une utilité : celle de fournir un rôle permettant d'y représenter les opérateurs de cardinalité des types d'éléments fils `Mail`, `Phone` et `Fax`.

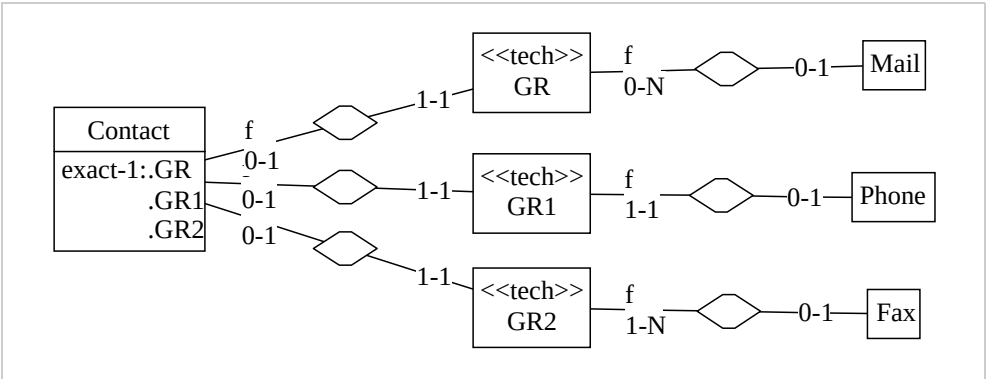

Figure 3.4- Les constituants forment un contenu de type `choice` et sont soumis à un opérateur de cardinalité

Au vu de la complexité de cette représentation, nous avons préféré définir une contrainte spécifique à notre cas : `choice` (et par homogénité la contrainte `seq`) et abandonner l'utilisation de la contrainte `exactly-1`. Ce choix, bien que privilégiant la lisibilité du schéma, exige une gestion plus importante car les contraintes `choice` et `seq` sortent du cadre du modèle Entité-Association Etendu.

3.3. Conclusion

Il est important de garder à l'esprit les deux remarques qui précèdent car elles soulignent des situations dans lesquelles la sémantique du DTD n'apparaît pas explicitement dans le schéma le représentant. Il est primordial d'en être conscient dans des processus tels que la conceptualisation d'un schéma XML.

4. Représentation d'une liste d'attributs

Tout attribut attaché à un type d'éléments est représenté par un attribut simple de même nom, contenu dans le type d'entités représentant ce type d'éléments.
On peut dénombrer six types d'attributs différents. Chacun d'eux a sa propre représentation que l'on va décrire dans la section suivante. La déclaration par défaut d'un attribut peut prendre quatre formes. Leur représentation sera décrite à la section 4.2.

4.1. Représentation du type d'un attribut

4.1.1. Attribut de type `ID` *ou* `IDREF`

Le mécanisme des attributs `ID/IDREF` s'apparente aux notions d'identifiant et de contrainte référentielle que l'on trouve dans le modèle GER, bien qu'il ait une portée plus large. C'est la raison pour laquelle, nous avons adopté une représentation similaire à celle adoptée pour les identifiants et les contraintes référentielles.
Un attribut de type `ID` est représenté en construisant un groupe constitué de l'attribut représentant cet attribut XML et soumis à l'action d'une contrainte spécifique nommée `gid` (abréviation de `global identifiant`). De manière similaire, un attribut de type `IDREF` est représenté par un groupe constitué de l'attribut représentant cet attribut XML et soumis à une contrainte spécifique nommée `idref`.
Si l'attribut est de type `IDREFS`, on procède de la même façon que pour le type `IDREF`, à la différence près que la cardinalité maximale de l'attribut représentant cet attribut XML vaut `N`.

Soit la déclaration d'une liste d'attribut attaché au type d'éléments `Employer` suivante :

```
<!ATTLIST Employer EmployerId ID #REQUIRED
                   Jobs IDREFS #IMPLIED >
```

Elle est représentée graphiquement par le schéma de la figure 4.1.

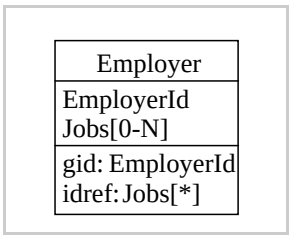

Figure 4.1- Représentation d'attributs de type ID et IDREF.

Le choix de la cardinalité minimale des attributs est décrit à la section 4.2.

4.1.2. Attribut de type `CDATA`

La déclaration `CDATA` signifie que la valeur prise par cet attribut est une chaîne de caractères prise littéralement, c'est-à-dire sans reconnaissance et traitement des caractères spéciaux de la syntaxe XML. L'attribut représentant cet attribut XML prend un type *user-defined*[1] nommé `cdata`.

1. Pour définir un type *user-defined* dans l'atelier DB-MAIN, il faut ouvrir la boîte de dialogue *Product/User domains ...* et cliquer sur le bouton *New.*

Par exemple, la déclaration suivante :

```
<!ATTLIST Employer website CDATA #IMPLIED>
```

sera représentée comme suit[1] :

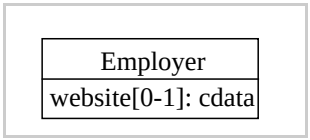

Figure 4.2- Représentation d'un attribut de type `cdata`.

4.1.3. Attribut de type `NMTOKEN`

La déclaration `NMTOKEN` signifie que la valeur prise par cet attribut est un nom symbolique constitué uniquement de caractères alphanumériques. Ce type d'attribut est représenté en associant à l'attribut représentant l'attribut XML, un type *user-defined* appelé `nmtoken`. Le type `NMTOKENS` est représenté de manière similaire au type `NMTOKEN` à la différence près que, dans ce cas, la cardinalité maximale de l'attribut représentant l'attribut XML vaut `N`.

4.1.4. Attribut de type `ENUMERATE`

La déclaration d'un attribut de type `ENUMERATE` est constituée d'une liste de valeurs séparées par des barres verticales. Ce sont les seules valeurs que peut prendre l'attribut. On utilise la méta-propriété `Value Constraint` de l'attribut représentant l'attribut XML pour les stocker.

La figure 4.3 illustre la valeur de la méta-propriété `Value Constraint` correspondante à la déclaration suivante :

```
<!ATTLIST Rapport language (English|French|Dutch|German|Other) "Other">
```

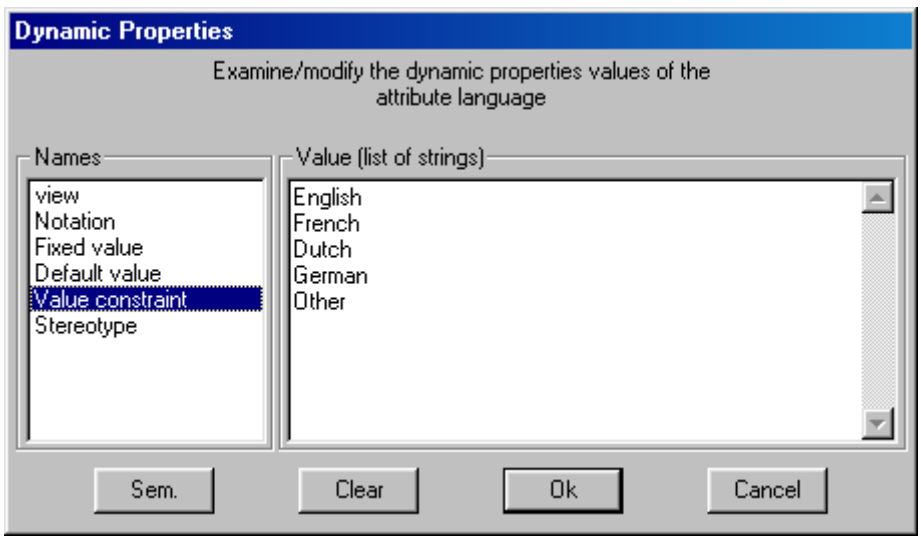

Figure 4.3- Les différentes valeurs que peut prendre un attribut de type `ENUMERATE` sont stockées dans la méta-propriété `Value Constraint`.

4.1.5. Attribut de type `NOTATION` *ou* `ENTITY`

Un attribut de type `NOTATION` (resp. `ENTITY`) doit prendre ses valeurs parmi l'ensemble des `NOTATION` (resp. `ENTITY non XML`) définies dans le DTD. Pour représenter ce type d'attributs, on définit

1. L'atelier DB-MAIN offre la possibilité d'afficher le type des attributs en cochant la case *Attributes types* du menu *View/Graphical Settings*.

un *user-domain* nommé `notation` (resp. `entity`).
Le cas des attributs de type `NOTATIONS` et `ENTITIES` se règle de la même façon que les types `NMTOKENS` et `IDREFS`, à savoir en affectant à l'attribut représentant l'attribut XML la cardinalité maximale `N`.

4.2. Représentation de la déclaration par défaut d'un attribut

Si l'attribut a une déclaration par défaut de la forme :

- `#REQUIRED`
  L'attribut représentant cet attribut XML a une cardinalité minimale égale à 1. C'est le cas, par exemple, de l'attribut `EmployerID` de la figure 4.1.

- `#IMPLIED`
  Dans ce cas, la cardinalité minimale affectée est nulle. C'est le cas, par exemple, de l'attribut `Jobs` de la figure 4.1.

- `#FIXED`
  Dans ce cas, l'attribut a une valeur fixée. Cette valeur apparaît dans la déclaration de l'attribut après le mot-clef `#FIXED`, entourée de guillemets. Elle est stockée dans la méta-propriété `Fixed Value` associée à l'attribut représentant cet attribut XML.

- `default`
  L'attribut a une valeur par défaut qui apparaît dans la déclaration de l'attribut directement après le type de l'attribut, entourée de guillemets. Elle est stockée dans la méta-propriété `Default Value` associée à l'attribut représentant l'attribut XML. La figure 4.4 montre l'état de la méta-propriété `Default Value` pour l'attribut `language` de la section 4.1.4.

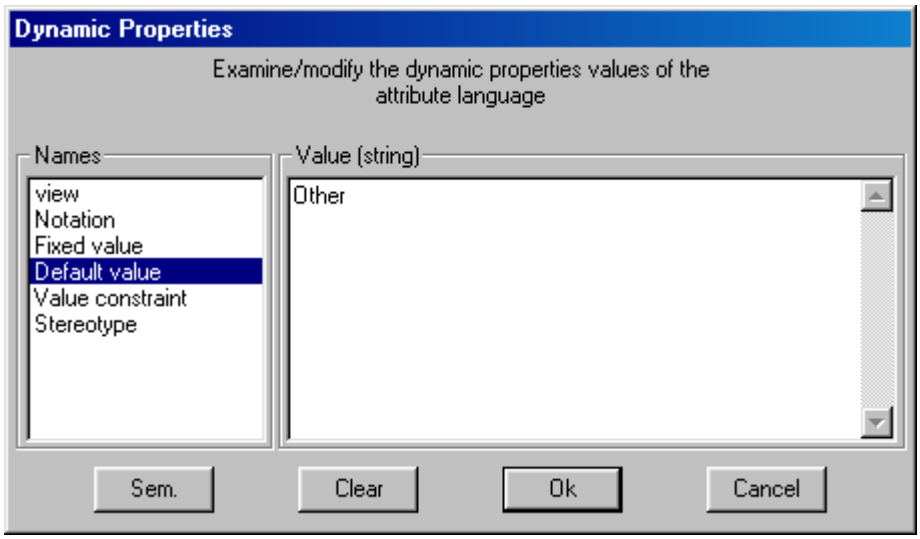

Figure 4.4- La valeur par défaut d'un attribut XML est stockée dans la méta-propriété `Default Value` asociée à l'attribut le représentant.

5. Déclaration de notations et d'entités

5.1. Déclaration d'entités

La représentation des entités ne porte que sur les entités générales. En effet, les entités paramètres ne sont que des raccourcis d'écriture dont l'usage est limité au DTD lui-même. On suppose que la représentation du DTD se fait après que toutes les références à des entités paramètres aient été remplacées par leur contenu.

Les déclarations d'entités générales sont stockées entièrement dans une méta-propriété nommée `Entity` définie sur le schéma lui-même.

Par exemple, si le DTD contient les déclarations suivantes :

```
<!ENTITY T "Technique">
<!ENTITY r2 SYSTEM ".../rapport2.doc" NDATA doc>
```

elles se retrouvent littéralement dans le contenu de la méta-propriété `Entity` attachée au schéma comme le montre la figure 5.1. La méta-propriété est multi-valuée afin de pouvoir stocker plusieurs déclarations.

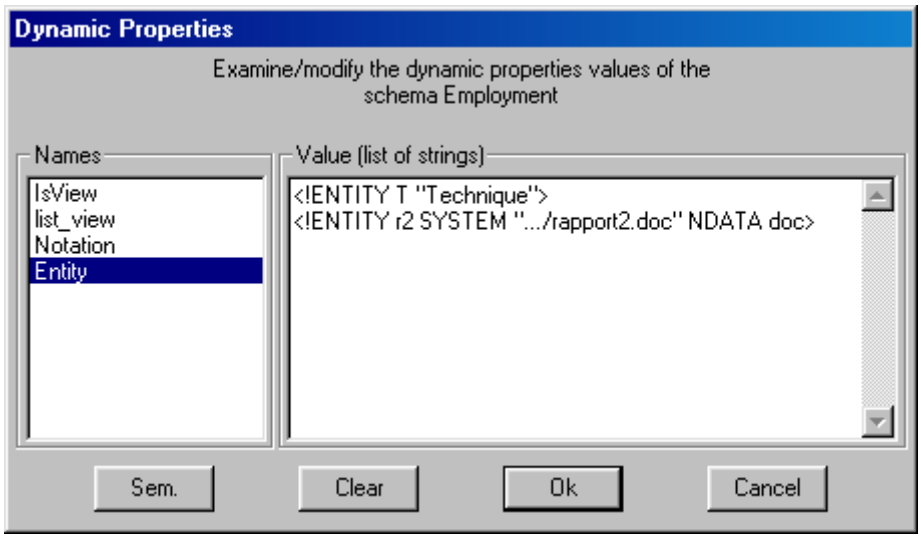

Figure 5.1- La méta-propriété `Entity` attachée au schéma contient les déclarations d'entités.

5.2. Déclarations de notations

De manière similaire aux déclarations d'`ENTITY`, les déclarations de `NOTATION` sont stockées littéralement dans le contenu d'une méta-propriété nommée `Notation` attachée au schéma.